Yugoslav Journal of Operations Research
27 (2017), Number , 31–45
DOI: 10.2298/YJOR150416013A

A HEURISTIC HYBRID FRAMEWORK FOR VECTOR JOB SCHEDULING

Nareyus I Lawrance AMALDASS
Brunel University, London, UK
nareyus22@gmail.com

Cormac LUCAS
Brunel University, London, UK
cormac.lucas@brunel.ac.uk

Nenad MLADENOVIC
LAMIH, University of Valenciennes, France and Mathematical Institute, SANU, Belgrade, Serbia
Nenad.Mladenovic@brunel.ac.uk

Received: April 2015 / Accepted: July 2016

Abstract: We examine the first phase of a known NP-hard 2-stage assembly problem. It consists of sequencing a set of jobs having multiple components to be processed. Each job has to be worked on independently on a specific machine. We consider these jobs to form a vector of tasks. Our objective is to schedule jobs on the particular machines in order to minimize the completion time before the second stage starts. We first develop a new mathematical programming formulation of the problem and test it on a small problem instance using an integer programming solver. Then, we develop a heuristic algorithm based on Ant Colony Optimization and Variable Neighborhood Search metaheuristics in order to minimize the total completion time. The performance of our implementation appears to be efficient and effective.

Keywords: Variable Neighborhood Search, Ant Colony Optimization, Scheduling, Integer Programming.

MSC: 90B06, 90C05, 90C08.

1. INTRODUCTION

The focus of this paper is on solving separately the first stage of the two stage assembly scheduling problems [2, 3] which various industries are faced with. For instance, consider the computer hardware industry. Computers are produced by assembling various components depending on customer specifications. These components are monitors, hard disks, CD/DVD ROM's, keyboards, mouse etc. Each components, having its own specifications, is made separately on an adequate machine. As computers are manufactured due to customers' specific requirements, a variety of computers is needed, which is managed by the combining the components in many different ways. When all the components are ready, they are sent to a client who does the second stage of assembling. Our objective is to minimize the waiting time of the client. The specific components are considered to be a vector of jobs which are independent and manufactured on a specific machine. The two stage models have been studied in Potts et al [3]. The results of Chen and Hall [1] give the lower bound on the performance guarantee, whereas in this paper we study upper bounds by using metaheruistic approaches. We minimize the customers waiting time while Potts et al [3] investigated the second stage objective function: minimize the makespan, including the time for assembling. The assembly departments are spread out and they are not implicitly included in our model. The completion time of a job is defined as the maximum time taken to complete the manufacturing all of its components. The problem is to schedule the vector of jobs such that the sum of the completion times of all the jobs is minimized. In this way, the assembling of all the final products will be completed as early as possible. We consider a deterministic formulation of the vector job scheduling problem. Chen and Hall [1] have already proved NP hardness of the problem. Therefore, it is natural to study Ant Colony Optimization (ACO) and Variable Neighborhood Search (VNS), general heuristic approaches, to solve this problem. We first describe the mathematical formulation of the problem. Then, we develop an ACO without local search, proposed by Dorigo and Carov [10], and implement VNS proposed by Mladenovic and Hansen [27]. We also develop a hybrid algorithm by combining ACO with VNS, as the local search, and implement our vector job scheduling problem. Finally we, compare our results with an integer programming solver.

This paper is organized as follows. In section 2, we review the relevant literature on this topic, while in section 3, we describe the mathematical model. Section 4, describe the hybrid algorithm (ACO and VNS). Our experimental results are discussed in section 5, and finally, the conclusions are given in Section 6.

2. LITERATURE REVIEW

Scheduling problems are high level synthesis problems, classified as computationally NP complete. To solve these problems means to determine the optimal time depending upon the objective function (job scheduling), environment, and characteristics of the jobs and the machines. Although we are concerned with

assembly line problems, as there are various type of scheduling problems, our literature survey gives different types of scheduling problems solved by using various algorithms. Hsu [4] has implemented an approximation algorithm for the assembly line crew scheduling problem in order to minimize the row sum by independently permuting the elements in the column. Chong et al [5] compared random generated populations and heuristic created populations with a genetic algorithm for assembly line balancing problems. Al-Anzi and Allahverdi [6] implemented a hybrid tabu search algorithm to minimize the completion time for two stage assembly problems. Webster and Azizgolu [7] used dynamic programming algorithms for scheduling parallel machines, incorporating a family setup time. In this approach, jobs are partitioned into families, where setup time is required for the first job of each family but not for the later jobs of the same family. Ishibuchi et al [8] implemented genetic algorithms with neighborhood search algorithms for fuzzy flow shop scheduling problems. Heinonen and Pettersson [14] solved job shop scheduling problems by using four different variants of the ACO algorithm. They further implemented a hybrid model which uses a postprocessing algorithm to improve the resulting schedule. Mladenovic et al [22] implemented a new variant variable neighborhood search, called 2 level general variable neighborhood search (GVNS), for solving travel salesman problems. They have used GNVS as a local search and show that it outperforms the tabu search heuristics. Seda [9] has constructed a mathematical model for flow and job shop problems, where he proposed different methods for small and large problem instances. The above literature shows that heuristic methods play a vital role in scheduling problems, where some provide good solutions and some provide good efficient computational time depending upon the problem size and the constraints. We consider ACO, VNS and hybrid(ACO and VNS) heuristic methods in order to compare and analyze their efficiency.

In recent years, researchers have shown that ACO, a meta-heuristic algorithm, is a competitive technique in terms of performance and time efficiency for various combinatorial optimization problems. ACO was initially proposed by Dorigo and Caro [10] and has been successfully implemented to solve many NP-problems, like the TSP [19], network routing, and quadratic programming problems. Furthermore, Dorigo and Stutzle [11] suggested various ACO approaches for quadratic assignment problems (QAP), e.g. ant system (AS), Rank based ant system (RAS), MAX-MIN Ant System (MMAS) and compared the results. Other researchers have implemented these ant based algorithms in various optimization problems like scheduling, vehicle routing, networking, option pricing, etc. Colorni et al [12] implement ACO for the job shop scheduling problem. Rajendran and Ziegler [13] proposed ACO for permutation flow shop scheduling to minimize makespan/total flow time of jobs. Huang and Liao [15] presented a hybrid algorithm, combining an ACO algorithm with a tabu search algorithm for the job shop problem to improve the solution quality. Zhang et al [20] implemented an ant system for a small job shop problem and compared it with traditional optimization methods. Eswaramurthy and Tamilarasi [16] proposed tabu search with ant colony optimization for job shop scheduling. They used dynamic tabu search strategies with

neighborhood search depending on the ant colony. Surekha and Sumathi [17] used a genetic algorithm and ant colony for solving fuzzy based job shop problems so that the sequence of jobs are scheduled using fuzzy logic and optimized using a genetic algorithm and ACO. Ponnambalam et al [18] proposed an ACO algorithm for flexible job shop scheduling problems that focus on scheduling tasks for the manufacturing industry in terms of improving machine utilization or reducing time.

Sevkli and Aydin [23] have proposed VNS for a job shop scheduling problem with makespan criterion. They have compared their results with the best known results in the literature and concluded that the VNS provides better quality solutions. Guohui at el [24] have proposed a hybrid algorithm, combining VNS with a genetic algorithm, for flexible job shop scheduling problems. They have used VNS to improve the quality of the individual in the GA by strengthening the local search ability. Liao and Cheng at el [25] proposed a new hybrid metaheuristic which uses tabu search within VNS to minimize the total weighted earliness and tardiness with a restrictive common due date in a single machine environment. They have examined 280 benchmark problem instances and showed that their algorithm produced a better quality solution and faster computational time. From this literature review we see that ACO is competitive with other metaheuristic approaches in terms of computational time and solution quality.

We consider the first stage of the two stage assembly scheduling problem where we use the ACO and VNS algorithm to solve this problem and compare the results with ACO,VNS and integer programming(CPLEX). We propose a hybrid algorithm, ACO with VNS. We further use VNS as the local search proposed within ACO. VNS uses shaking and local search neighborhood techniques.

3. MATHEMATICAL PROGRAMMING FORMULATION

The vector job shop problem (VJS) is to schedule the jobs on the corresponding machines so that the time of the assembly machine to process the jobs is minimized. The processing jobs should remain in the same sequence on each machine and no job is allowed to be interrupted on the machine. The mathematical programming formulation is given below.

Indices

- $J = \{1, 2, \ldots, j, \ldots, n\}$ denotes a set of jobs,
- $M = \{1, 2, .., i, ., m\}$ denotes a set of machines.

Data

- t_{ij} be the time spent on the machine i by job j.

Variables

- C_j be the time when job j is completed,

- x_{jk} a binary variable that gets value 1 if job j is performed before job k.

$$x_{jk} = \begin{cases} 1 & \text{if job } j \text{ is before } k \\ 0 & \text{otherwise.} \end{cases}$$

The complete mathematical programming formulation can be written as:

$$min \sum_{j=1}^{n} C_j \tag{1}$$

subject to

$$C_j \geq t_{ij} + \sum_{k=1 k \neq j}^{n} t_{ik} x_{kj}, \ \forall\, j \in J, \forall\, i \in M \tag{2}$$

$$x_{jk} + x_{kj} = 1, \ \forall\, j,k \in J, \ j \neq k \tag{3}$$

$$x_{jk} + x_{kl} + x_{lj} \leq 2, \ \forall\ j,k,l \in J, \ j \neq l \neq k \tag{4}$$

$$x_{jk} \in \{0,1\}, \forall\, j,k \in J. \tag{5}$$

The objective function is to minimize the total completion time. The constraint group (2) determines the earliest completion time of each job by considering the time of its completion on every machine. The summation term ensures that all earlier tasks on this machine are completed before this task. The constraint group (3) ensures that job j is performed always before job k, or job k is completed before job j. In case that job j is before job k then, it must be true for all machines. The constraint group (4) is a cycle breaking constraint, i.e., it ensures that the ordering of jobs doesn't fall into a cycle. Thus, if job j is before job k and job k is before job l, then job j must be before job l.

4. HYBRID ALGORITHM FOR VJS

For solving the VJS problem in this section we first present two metaheuristic based approaches, ACO and VNS and then we propose a hybrid heuristic which combines them.

4.1. ACO

The algorithm (ACO) [10] is a metaheuristic inspired by behavior and characteristics of ants. In general, real ants deposit a chemical substance called pheromone λ while travelling in search for food. Pheromone acts as a scent to attract the ant back along the same path when returning to the nest. Initially, the pheromone value is set to zero. As the ants travel in search for food, pheromone is deposited along the path so that other ants are attracted this path. During the course of time, more ants get attracted to the shortest path, since on the longer

paths the pheromone tends to evaporate and ants find it to be an unproductive path. We use this approach in our vector job scheduling problem.

In our VJS problem, we establish the sequence of jobs in order to minimize the completion time. We could formulate the VJS problem by constructing an edge between the jobs only if the corresponding job can be performed by the corresponding machine component. This process could be modelled in the form of a matrix. Table 1 is a sample of the constructed matrix with m machines and n jobs, where the rows represent the machines and the columns represent the jobs. Each matrix cell represents the processing time t_{ij}. Our objective is to sequence the jobs in such a way that the total processing time C is minimized.

Table 1: m machines and n jobs

Machine/Job	1	2	3	4	..	j	..	n
1	t_{11}	t_{12}	t_{13}	t_{14}	..	t_{1j}	..	t_{1n}
2	t_{21}	t_{22}	t_{23}	t_{24}	..	t_{2j}	..	t_{2n}
3	t_{31}	t_{32}	t_{33}	t_{34}	..	t_{3j}	..	t_{3n}
4	t_{41}	t_{42}	t_{43}	t_{44}	..	t_{4j}	..	t_{4n}
.	..	..	..	..	..	$t..$	..	..
i	t_{i1}	t_{i2}	t_{i3}	t_{i4}	..	t_{ij}	..	t_{in}
.	..	..	..	..	..	$t..$	..	..
m	t_{m1}	t_{m2}	t_{m3}	t_{m4}	..	t_{mj}	..	t_{mn}

4.1.1. Solution Construction

We construct a disjunctive graph $G = (N, E)$ where N is the set of nodes that represent processing times t_{ij}, job j on machine i. The initial node N_0 and the final node N_{n+1} are considered as dummy nodes that are used to indicate the start and the end node for the ants. E is a set of edges, each edge has a measure associated with the amount of pheromone λ scent deposited while the heuristic information is the inverse of processing time on the nodes. In figure 1, we consider 3 jobs sets J_1, J_2, J_3 to be processed on a machines i_1. Each job set consists of permutation of jobs $j'1, j'2, j'3$. The circle represents the processing time t_{ij} in which a job j is performed on a machine i. An ant k starts from the initial node, finds its own path and creates a solution by forming a sequence of visiting nodes. We have to verify that the ants do not violate any of the constraints such as maintaining the correct job ordering on each machine. The visiting of nodes is in sequential ordering. Each path on the graph from node N0 to node N10 which does not violate any of the constraints represents one permutation of jobs, i.e. one solution of the problem. Thus if the first node is job j_1 on machine i_1 then, the ant k might have a

choice of choosing j_2 or j_3 on machine i_1, hence allocating all corresponding jobs in machine i_1. The same job sequence is followed on all other machines.

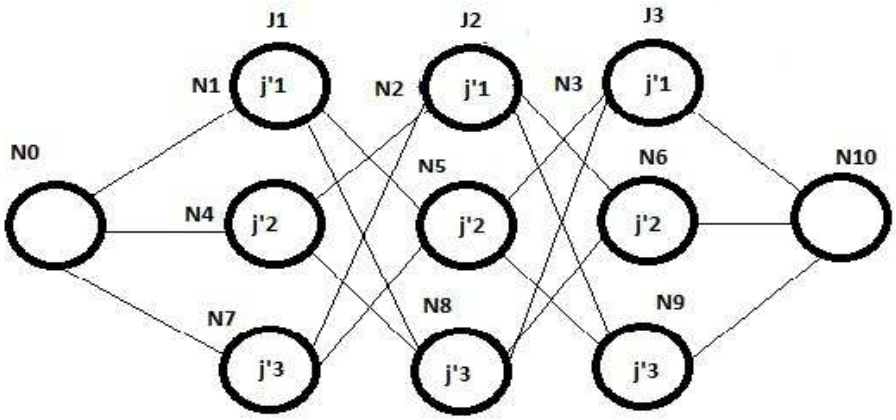

Figure 1: 3 set job sequence for machine 1(i1)

Initially, the memory is empty and is updated according to the ants movements through different nodes. The ant memory can be separated into three parts. Part 1 is the set of nodes not yet visited. Part 2 consists of the set of nodes that the ants are able to visit in the current iteration in order to satisfy the constraints, and finally, part 3 contains the set of all nodes that have been visited so far. Hence when the ant is placed at the initial node it travels accordingly after completion of one cycle, each ant should have a sequence of nodes in group 3 that satisfies the constraints of visiting one node only and the job machine ordering constraints' too. The parameters for ACO are as follows.

- k - the number of Ants,
- $\lambda(x, y)$ - the pheromone deposit on the edge (x, y),
- ρ - the pheromone evaporation parameter,
- α - the level of pheromone scent deposited by the current ants,
- β - the heuristic information determined by the ants,
- $\eta(y)$ - the heuristic information stored at node y,
- N - the set of all nodes,
- V - the set of visited nodes,

- U - the set of unvisited nodes.

Starting from the initial node, each ant chooses the next node by using the transition probability rule, which uses the amount of pheromone deposited on its path and the heuristic information. Given U is the set of unvisited nodes, let us consider an ant that has travelled to node x. The probability of choosing the next node y is given by

$$P_k(x,y) = \begin{cases} \frac{[\lambda(x,y)]^{\alpha}.\eta(y)^{\beta}}{\sum_{y \epsilon U}[\lambda(x,y)]^{\alpha}.\eta(y)^{\beta}} & \text{if } y \epsilon U, \\ 0 & \text{otherwise.} \end{cases}$$

where $P_k(x,y)$ is the probability with which ant k chooses to move from node x to node y; $\eta(y) = 1/p(y)$, which is the inverse of the processing time of operations y, where $p(y) = t_{ij}$ and y represents job j on machine i.

4.1.2. Pheromone Update

Once the ant completes the tour we use a procedure called pheromone update. The pheromone trail updates globally and locally. A global pheromone update, focuses on edges belonging to the trail travelled by ant within the shortest time. Once the time is completed, the best ant deposits pheromone on visited edges, while the other edges remain unchanged. The amount of pheromone deposited, $\Delta\ \lambda(x,y)$, on each visited edge (x,y) by the best ant is inversely proportional to the length of the time of the tour. The shorter the time, the greater the amount of pheromone deposited on the edges. The global updating of pheromone is given by

$$\lambda(x,y) = (1-\rho)\lambda(x,y) + \rho\Delta\lambda(x,y)^{best}$$

where,

$$\Delta\lambda(x,y)^{best} = \begin{cases} \frac{1}{L_{best}} & \text{if the best ant uses the edge (x,y) in its sequence,} \\ 0 & \text{otherwise.} \end{cases}$$

ρ is the evaporating parameter, $0 < \rho < 1$. L_{best} is the length of the best sequence in the current iteration. Hence the best sequence determines the amount of pheromone deposited in each iteration. Searching continues while the current pheromone level is below the previous value.

The local pheromone update is focuss to avoiding a strong edge being chosen by all the ants. This is done by changing the amount of pheromone update when an edge is chosen by a specific ant. The local update is given by

$$\Delta\lambda(x,y) = (1-\rho)\lambda(x,y) + \rho\Delta\lambda_0$$

where $\Delta\lambda_0$ is a parameter which is set to 1.

Algorithm 1: ACO

```
Initialize all edges to pheromone level λ0 = 1;
Place each ant k on a randomly chosen job j on a particular machine i;
for each iteration t do
    while each ant k has not completed its set of jobs J do
        for each ant k do
            Move ant k to the next job j by the probability function Pk;
        end
        for each ant k with a complete set of jobs J do
            Evaporate pheromone ρ;
            Apply pheromone updates;
            if ant k time is the shortest then
                global pheromone update;
                Update global solution;
            end
        end
    end
end
```

4.2. Variable Neighborhood Search (VNS)

VNS is a metaheuristic that explores distant neighborhood structures of growing size to identify better local optima with local search strategies [27]. During the local search, VNS systematically changes different neighborhoods. The VNS is based on three key factors namely, (i) A local optimal solution of one neighborhood structure is not necessary to be that of another neighborhood structure; (ii) A global optimal solution is a local minimum with respect to all neighborhood structures; (iii) Local optimum solutions with respect to different neighborhoods are relatively close to each other. For more details about VNS and its applications see e.g. [28, 29, 30, 31].

In our implementation, the solution of the problem is represented by two arrays. (i) A sequence of jobs $j_1, j_2, ..., j_n$; (ii) The sequence of corresponding completion time, C_j. Therefore we present a solution S as $S = (\acute{J}, C)$. Our proposed algorithm is based on general VNS that comprises of shaking and local search techniques. The proposed algorithm uses 3 neighborhood structures, the first is the 1-opt moves, the second is the insertion, and the third is the swap. Below is the description of all these operations. Let S be the initial solution and $\ddot{S}$ the new solution after applying the respective neighborhoods, then

- *1-Opt neighborhood*: A new solution is reached by interchanging two consecutive elements in the current array $(\acute{J}, C)$.
- *Insert neighborhood*: A new sequence is generated by inserting one job between any two other jobs. For example the job 7 is inserted between jobs 4 and 5 to get a new sequence of jobs, $\acute{J}$.

Initial solution	3	4	5	6	7	1

After Insert	3	4	7	5	6	1

- *Swap neighborhood structure*: A new sequence is generated by interchanging the position of jobs in the sequence of jobs $\acute{J}$. For example, job 6 in the 4^{th} position is interchanged with job 3 which is in the 1^{st} position.

Initial Solution	3	4	5	6	7	1

The swap neighborhood structure is used in the shaking step to improve the initial solution S i.e., neighborhood $N_k(S)$ is defined by consecutive swaps of jobs $\acute{J}$ followed by updating its corresponding values of C_j to get a better solution $\dot{S}$. In order to improve the solution $\dot{S}$, we use a local search step, consisting of a one opt neighborhood, by changing the consecutive jobs $\acute{J}$; then, we change the neighborhood to insertion by inserting job j_i between any two jobs j_u, j_v in the job sequence $\acute{J}$. At each neighborhood, a solution $\ddot{S}$ is generated and the best solution, if improved, is updated. The process of VNS pseudo code is presented below. The shaking is executed in step 4 with local search steps in 5-16, which are used to improve the solution; n represents the maximum number of loops.

Algorithm 2: VNS

```
1  Generate an initial solution S randomly;
2  Iteration t ← 0;
3  do
4  |  Ṡ = Swap(S);
5  |  loop ← 0;
6  |  do
7  |  |  count ← 0;
8  |  |  max ← 2;
9  |  |  do
10 |  |  |  if count ← 0 then S̈ = Swap(Ṡ);
11 |  |  |  if count ← 1 then S̈ = Insert(one − Opt(Ṡ));
12 |  |  |  if f(S̈) ≤ f(Ṡ) then count ← 0; Ṡ ← S̈ ;
13 |  |  |  else count + +;
14 |  |  while count < max;
15 |  |  loop + +;
16 |  while loop < n(n − 1);
17 |  if (f(Ṡ) ≤ f(S)) then S ← Ṡ ;
18 |  t + +;
19 while stopping condition is not met;
```

After Swap	6	4	5	3	7	1

4.3. Hybrid Algorithm

In the previous section, the proposed VNS algorithm randomly generated the initial solution. This may not be an appropriate way to solve large problem instance as it might take more computational time to converge to an optimal solution. In our proposed hybrid approach, we use the ACO algorithm to generate the initial solution and then VNS to find a better solution, which will improve the performance. Our hybrid approach combines ACO and VNS. This approach is the same as the VNS algorithm except that the initial solution is generated by the ACO algorithm.

Algorithm 3: Hybrid ACO+VNS

```
1  Generate solution from ACO algorithm 1;
2  Iteration t ← 0;
3  do
4      Ṡ = Swap(S);
5      loop ← 0;
6      do
7          count ← 0;
8          max ← 2;
9          do
10             if count ← 0 then S̈ = Swap(Ṡ);
11             if count ← 1 then S̈ = Insert(one − Opt(Ṡ));
12             if f(S̈) ≤ f(Ṡ) then count ← 0; Ṡ ← S̈ ;
13             else count + +;
14         while count < max;
15         loop + +;
16     while loop < n(n − 1);
17     if (f(Ṡ) ≤ f(S)) then S ← Ṡ ;
18     t + +;
19 while stopping condition is not met;
```

5. COMPUTATIONAL RESULTS

We have used an Intel core dual i7 processor, 3.4GHz with windows 7, 64-bit operating system with 16GB RAM. The ACO and VNS have been developed in the *C#* language. Since our problem is new, we are unable to find any benchmark instances for it, we have implemented our algorithms on 15 various well known benchmark instances from the literature that were not originally designed for solving VJS problem.

We have used the following test instances:

- Fisher and Thompson job instances for 10x10 and 6x6 [32];

- Lawrence instances for 15x15, 30x10, 15x10, 20x5 [33];
- Storer, Wu and Vaccari job instances for 20x10, 20x15, 50x10 size problems [34];
- Talliard flow shop instances for dimensions 100x5, 100x10, 100x20, 200x10, 500x20 [35];
- Yamada and Nakano for job instances of 20x20 [36].

For each instance, the stopping criteria for ACO and VNS was set to 1000 iterations. The best solutions are reported in Table 2. The following section contains the parameter setting for the ant system and the results discussion.

5.1. Setting parameter values for ACO

The choice of parameter values of the ACO algorithm play an important role in the final solution quality. The ACO algorithm literature proposed by Liouane et al [21] reviews various parameter values. In VJS we set various parameter values accordingly to obtain the best solution. We only consider the values in the range detailed in [21].

- α determines the quantity of pheromone deposited by the ants when they build their solution. The higher the value, the more restrictive the ants ability becomes in exploring new sequences. The lower the value, the more the ant is able to travel to more unvisited edges, but this uses more computational time. From our analysis, we have found the value to be in the range 1-10 and overall the algorithm performs best with a value of approximately 1.
- β determines the heuristic information used by the ants. The value ranges between 1- 4. From our results we have observed that the value 1 gives the best results.
- ρ is the pheromone evaporation parameter. A higher value enables the ants to carry out more searches. It has been noted that the value ranges between 0.1 - 0.7 for obtaining good solutions. We have observed that 0.7 is the best value.
- k defines the number of ants belonging to the colony. There is a trade-off between creating too many sequences, namely too many ants, versus fewer sequences namely fewer ants. If the number of ants is low, the algorithm speeds up because of fewer searches; in contrast, if the number of ants is high, the algorithm slowdowns because of more search, but many more sequences are created with the scope of finding a better solution. Our experiments suggest that the number of ants in the colony is best to be between the ranges of 8 - 10.We have used 10 ants for our entire job instance.

5.2. Results

Table 2 contains the results. We have computed the solutions for ACO, VNS, ACO+VNS and CPLEX. Three heuristic methods are run 1000 times and the best solution obtained is reported in the table. The computing time for 1000 iteration of each heuristic method and the total CPU time(seconds) are given in the brackets. The following conclusion can be derived:

Table 2: Best solutions for VSJ instances

Instance	ACO	VNS	ACO+VNS	CPLEX
[20] × [5]	11392(331)	10883(9654)	10883(9844)	10883(46020)
[10] × [10]	3211(30)	3205(9321)	3205(9735)	3205(1170)
[15] × [10]	7637 (35)	7530 (9638)	7530(9975)	7530(25522)
[15] × [15]	6958(96)	6870(9644)	6870(9980)	6870(31278)
[20] × [10]	11794(238)	11341(9875)	11341(10677)	11341(2067290)
[20] × [15]	12402(351)	12021(9967)	12021(10831)	NA
[20] × [20]	6664(181)	6486(9998)	6486(11456)	NA
[30] × [10]	24824(461)	23268(14771)	23268(16543)	NA
[50] × [10]	68243(2538)	62841(65244)	62781(68022)	NA
[100] × [5]	263505(13205)	233971(302640)	233944(351108)	NA
[100] × [20]	267499(20193)	242959(1041607)	242930 (1053421)	NA
[100] × [10]	274356(16010)	248109(566441)	248025(598007)	NA
[200] × [10]	1056487(122576)	931889(4930348)	931429(5035668)	NA
[500] × [20]	6418210(672757)	5847759(19968785)	5846398(20851539)	NA

(i) In terms of solution quality, it is clear that ACO + VNS performs better than the other algorithms, especially for larger problem instances. However, it uses more computational effort than our general VNS algorithm;

(ii) CPLEX is unable to find results for large instances;

(iii) VNS outperforms ACO in all instances significantly, but uses more cpu time.

6. CONCLUSION

In this paper, we observed that the first phase of the known assembly scheduling problem [3] may have its own practical implementation, i.e., when the second assembly phase is omitted. We call the first phase, proposed by Patkar et al [26] a vector scheduling problem, we have proposed a new mathematical programming formulation and solved it with a commercial solver CPLEX. Further, we have presented four different types of computational techniques to solve the VJS problem: (i) an exact method based on our mathematical programming formulation; (ii) a heuristic method based on ACO meta heuristic; (iii) a heuristic method based on VNS and finally (iv) as hybrid of ACO that uses VNS as the local search method. Our goal was to identify the performance of these techniques for the vector job scheduling problem. The performance analyzes were carried out by using the proposed method with our data sets and the results were compared. Our results show that the ant colony algorithm with variable neighborhood search as the local

search finds high quality solution to the VJS or at least comparable to the other algorithms proposed for the VJS, particularly as the size of the problem instance is increased. Even though ACO with VNS performs well, it depends upon setting the right parameter values and selecting the number of ants for each problem. From our analysis we have shown that the pheromone trail updates between the nodes plays a vital role in constructing the best solution.

Future work may include development of more advanced VNS based techniques such as skewed general VNS. Another task could be investigation in finding more applications of VJS model.

Acknowledgement: The work by Nenad Mladenovic was conducted at National Research University, Higher School of Economics, Russia and supported by RSF grant 14-41-00039.

REFERENCES

[1] Chen, Z.L., and Hall, N.G., "Supply chain scheduling conflict and cooperaion in assembly systems", *Operation Research*, 55 (6) (2007) 1072-1089.

[2] Potts, C.N., "An algorithm for the single machine sequencing problem with precedence constraints". *Mathematical Programming Studies*, 13 (1980) 78-87.

[3] Potts, C.N., Sevastjanov, S.V., Strusevich, V.A., Van Wassenhove, L.N., and Zwaneveld, C.M., "Two stage assembly scheduling problem:complexity and approximation", *Operation Research*, 43 (2) (1995) 346-355.

[4] Hsu, W.N., "Approximation algorithms for the assembly line crew scheduling problem", *Mathematics of Operation Research*, 9 (3) (1984) 376-383.

[5] Chong K.E., Omar, M.K., and Bakar, N.A., "Solving assembly line balancing problem using genetic algorithm with heuristics treated initial population", *Proceedings of The World Congress on Engineering*, 2 (2008) 1273-1277.

[6] Al-Anzi, F.S., and Allahverdi, A., "A hybrid tabu search heuristic for the two-stage assembly scheduling problem", *International Journal of Operations Research*, 3 (2) (2006) 106-119.

[7] Webster, S., and Azizgolu, M., "Dynamic programming algorithm for scheduling parallel machines with family setup time", *Computer and Operation Research*, 28 (2) (2001) 127-137.

[8] Ishibuchi, H., Yamamoto, N., Murata, T., and Tanaka, H., "Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems", *Fuzzy Sets and Systems*, 67 (1) (1994) 81-100.

[9] Seda, M., "Mathematical model for flow job and job shop problems", *World Academy of Science, Engineering and Technology*, 18 (3) (2007) 331-342.

[10] Dorigo, M., and Caro, D.G., "The ant colony optimization meta-heuristic", *New Ideas in Optimization, McGraw-Hill, London, UK*, (1999) 11-22.

[11] Dorigo, M., and Stutzle, T., "Ant colony algorithms for quadratic assignment problem", *New Ideas in Optimization, McGraw-Hill, London, UK*, 1999.

[12] Colorni, A., Dorigo, M., Maniezzo, V., and Trubian, M., "Ant system for job shop scheduling", *Belgian Journal of Operations Research, Statistics and Computer Science*, 34 (1) (1994) 39-53.

[13] Rajendran, C., and Ziegler, H., "Ant colony algorithms for permutation flowshop scheduling to minimize makespan total flowtime of jobs", *European Journal of Operational Research*, 155 (2) (2004) 426-438.

[14] Heinonen, J., and Pettersson, F., "Job-shop scheduling and visibility studies with a hybrid ACO algorithm", *Applied Mathematics and Computation*, 187 (3) (2007) 989-998.

[15] Huang, K., and Liao, C., "Ant colony optimization combined with tabu search for the job shop scheduling problem", *Computers and Operations Research*, 35 (4) (2008) 1038-1046.

[16] Eswaramurthy, V.P., and Tamilarasi, A., "Hybridizing tabu search with ant colony optimization for solving job shop scheduling problems", *International Journal of Advance Manufacturing Technololgy*, 40 (2008) 1004-1015.

[17] Surekha, P., and Sumathi, S., "Solving fuzzy based job shop scheduling problems using GA and ACO", *Journal of Emerging Trends in Computing and Information Sciences*, 1 (2) (2010) 95-102.
[18] Ponnambalam, S. G., N. Jawahar, N., and Girish, B. S., "An ant colony optimization algorithm for flexible job shop scheduling problem", *New Advanced Technologies*, 4 (2010) 73-94.
[19] Dorigo, M., and Gambardella, L. M., "Ant colonies for the traveling salesman problem", *Biosystems*, 43 (1997) 73-81.
[20] Zhang,J., Hu,X., Tan,X., Zhong, J.H., and Huang, Q., "Implementation of an ant colony optimization technique for job shop scheduling problem", *Transactions of the Institute of Measurement and Control*, 28 (1) (2006) 93-108.
[21] Liouane, N., Saad, I., Hammadi, S., and Borne, P., "Ant systems and local search optimization for flexible job shop scheduling production", *International Journal of Computers, Communications and Control*, 2 (2) (2007) 174-184.
[22] Mladenovic, N., Todosijevic, R., and Urosevic, D., "Two level general variable neighborhood search for attractive traveling salesman problem", *Yugoslav Journal of Operations Research*, 23 (1) (2012) 19-30.
[23] Sevkli, M., and Aydin, M., "Variable neighborhood search algorithm for job shop scheduling problems", *Evolutionary Computation in Combinatorial Optimization*, 3906 (2006) 261-271.
[24] Zhang, G., Gao, L., Li, X., and Li, P., "Variable neighborhood genetic algorithm for the flexible job shop scheduling problem", *Intelligent Robotics and Applications*, 5315 (2008) 503-512.
[25] Liao, C., and Cheng, C., "Variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date", *Computers and Industrial Engineering*, 52 (4) (2007) 404-413.
[26] Patkar, S., Poojari, C., and Porwal, P., "An investigation into approximate solutions for deterministic and stochastic multi-dimensional sequencing", *Brunel University*, Archive, Brunel, 2005.
[27] Mladenovic, N., and Hansen, P., "Variable neighborhood Search", *Computers and Operations Research*, 24 (11) (1997) 1097-1100.
[28] Hansen, P., Mladenovic, N., Brimberg, J., and Moreno Prez, JA., "Variable neighbourhood search, Handbook of Metaheuristics, 2nd edition (Gendreau and Potvin Eds)", *International Series in Operations Research & Management Sciences*, 146, Kluwer, (2010) 61–86.
[29] Brimberg, J., Hansen, P., and Mladenovic, N., "Attraction probabilities in variable neighborhood search",*4OR*, 8 (2010) 181-194.
[30] Mladenovic, N., Urosevic, D., and Dionisio Perez-Brito., "Variable neighborhood search for Minimum Linear Arrangement Problem", *Yugoslav Journal of Operations Research*, 26 (1) (2016) 3-16.
[31] Todosijevic, R., Hanafi, S., Lazic, J., and Mladenovic, N., "Variable and Single Neighbourhood Diving for MIP Feasibility", *Yugoslav Journal of Operations Research*, 26 (2) (2016) 131-157.
[32] Fisher, H., and Thompson, G.L., " Probabilistic learning combinations of local job-shop scheduling rules, J.F. Muth, G.L. Thompson (eds.)", *Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey*, 1963.
[33] Lawrence, S.," Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)", Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania, 1984.
[34] Storer, R.H., Wu, S.D., and Vaccari, R.,"New search spaces for sequencing instances with application to job shop scheduling", *Management Science*, 38 (1992) 1495-1509.
[35] Taillard, E., "Benchmarks for basic scheduling problems", *European Journal of Operational Research*, 64 (1993) 278-285.
[36] Yamada, T., and Nakano, R.,"A genetic algorithm applicable to large-scale job-shop instances, R. Manner, B. Manderick (eds.)", *Parallel instance solving from nature 2*, North-Holland, Amsterdam, 1992, 281-290.